STŘEDNÍ PRŮMYSLOVÁ ŠKOLA  
MLADÁ BOLESLAV

**ROČNÍKOVÁ PRÁCE**

Tomáš Hudeček

Mladá Boleslav 2022

STŘEDNÍ PRŮMYSLOVÁ ŠKOLA  
MLADÁ BOLESLAV

**ROČNÍKOVÁ PRÁCE**

**Autor: Tomáš Hudeček  
Studijní obor: 18-20-M/01 Informační technologie  
Vedoucí práce: Jan Till**

Mladá Boleslav 2022

# Obsah

[Obsah 3](#_Toc102065670)

[1 Úvod 6](#_Toc102065671)

[2 Discord bot 7](#_Toc102065672)

[2.1 Vytvoření a spuštění 7](#_Toc102065673)

[2.2 Komunikace se serverem 7](#_Toc102065674)

[2.3 Příkazy 8](#_Toc102065675)

[2.3.1 Čtení příkazů 8](#_Toc102065676)

[2.3.2 Funkcionalita příkazu 9](#_Toc102065677)

[2.3.3 Jednotlivé příkazy 11](#_Toc102065678)

[2.3.4 Zaregistrování příkazů na server 13](#_Toc102065679)

[3 Minecraft plugin 14](#_Toc102065680)

[3.1 Vytvoření a nasazení 14](#_Toc102065681)

[3.2 Komunikace s klientem 14](#_Toc102065682)

[3.3 Koncové body 15](#_Toc102065683)

[3.3.1 Get online players 16](#_Toc102065684)

[3.3.2 Give all cookie 16](#_Toc102065685)

[3.3.3 Give player cookies 16](#_Toc102065686)

[3.3.4 Kill all 16](#_Toc102065687)

[3.3.5 Kill 16](#_Toc102065688)

[3.3.6 Message 17](#_Toc102065689)

[3.3.7 Starve all 17](#_Toc102065690)

[3.3.8 Starve 17](#_Toc102065691)

[4 Přílohy 17](#_Toc102065692)

[4.1 Seznam obrázků 17](#_Toc102065693)

[4.2 Zdroje 18](#_Toc102065694)

**Prohlášení**

Prohlašuji, že jsem svou ročníkovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

Nemám závažný důvod proti zpřístupňování této ročníkové práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V Mladé Boleslavi dne podpis:

# Úvod

Smyslem této práce bylo vytvořit systém který je schopen propojit Discord[[1]](#footnote-1), což je komunikační platforma pro komunity, se známou hrou s otevřeným světem Minecraft[[2]](#footnote-2). Hlavním účelem bylo dát možnost uživateli na Discordu interagovat se serverem, který běží na Minecraftu.

Takový server si lze zakoupit online, nebo nahostovat[[3]](#footnote-3) na domácím počítači nebo notebooku. Do Minecraft serveru lze přidat plugin[[4]](#footnote-4), který využívají programátoři pro změnu designu a mechaniky her, dává jim veškerou moc manipulovat se serverem na kterém je plugin vložen.

Discord je, jak už je zmíněno, komunikační platforma, na kterém si může uživatel vytvořit server. Na tento server se mohou napojit další uživatelé. V takovém serveru se nacházejí kanály, kde si mohou členové psát nebo spolu volat. Na server si může vlastník napojit takzvaného Discord bota, což je aplikace řízená umělou inteligencí, která je schopná všeho na co je naprogramovaná.

Vytvořením těchto dvou aplikací, které spolu navzájem komunikují je zajištěno správné fungování systému.

Principem systému je, že bot vytvoří příkazy, které pak může člen na Discord serveru, kde je bot vložen, napsat. Tím pošle žádost na propojený plugin na Minecraft serveru pro zpuštění příslušného události.

# Discord bot

Zvolený softvérový systém pro vytvoření a spuštění bota byl zvolen Node.js a psaný byl v jazyce javascript, ve Visual studio code. Bot byl pojmenován McBot.

## Vytvoření a spuštění

Pro vytvoření bota je třeba vytvořit novou aplikaci na oficiálním developer portálu Discordu[[5]](#footnote-5). Každý bot má svůj unikátní token, díky kterému lze bota dále naprogramovat.

require('dotenv').config();

**const** { Client } = require('discord.js');

**const** client = **new** Client({});

client.login(process.env.TOKEN);

Zdrojový kód č. 1: Vytvoření a zpuštění bota

bot je v tomto zdrojovém kódu Client na kterého se pak volá login který zajistí spuštění bota.

procces.env.TOKEN je token bota, je ale uložen v .env souboru jelikož je nutné mít token bezpečně někde uložen. Env soubor se čte pomocí dotenv balíčku.

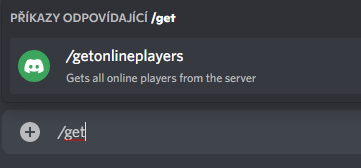
Bot se v základu spouští pomocí příkazu “node index.js“. V této aplikaci je ale využit baliček nodemon, který umožňuje automatické spuštění aplikace po změně kódu tedy “npm start“.

## Komunikace se serverem

Komunikace mezi klientem a serverem probíhá na http protokolu. Discord bot se chová jako klient a Minecraft plugin jako http server. Komunikaci s http serverem byl v botovy použit Axios[[6]](#footnote-6) balíček. Balíček byl využit pro jednoduché vytváření http požadavek.

## Příkazy

Uživatel napíše do kolonky pro zprávu název daného příkazu



Obrázek 1: Psaní příkazu

Po odeslání příkazu pošle bot http požadavek na koncový bod souvisící s daným příkazem. Server buď odpoví se zprávou, že se funkce pod daným koncovým bodem provedla správně či nikoliv. Při špatném zadání příkazu bot vrátí odpověď “Wrong command”.



Obrázek 2: Chyba na straně aplikace



Obrázek 3: Špatně zadaný příkaz

### Čtení příkazů

Bot naslouchá Discord chatu, čeká na zpuštění příkazu a jakmile ho někdo napsal, tak kontroluje, zda příkaz nezadal jiný bot, to by totiž mohlo způsobit cykli a další chyby.

client.on('interactionCreate', async interaction => {

**if** (!interaction.isCommand()) **return**;

**if** (!interaction.user.bot) **return**;

Zdrojový kód č. 2: Naslouchání příkazu

Potom jestli příkaz vůbec existuje a na konec vykoná funkci k němu řazenou.

**const** command = client.commands.get(interaction.commandName);

**if** (!command) **return**;

//…

command.execute(interaction);

Zdrojový kód č. 3: Zjištění a vykonání interakce

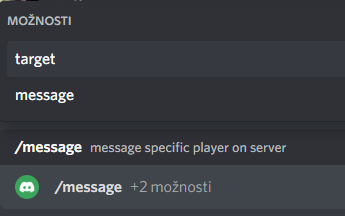
### Funkcionalita příkazu

Příkazy jsou uložené ve složce commands. A každý jednotliví příkaz je napsán v samostatném souboru.



Obrázek 4: Struktura příkazů

Každý byl vytvořen pomocí exportování modulu SlashCommnadBuilder[[7]](#footnote-7). Exportuje se do spouštěcího souboru index.js, ve kterém se na základě toho zjistí, zda napsaný příkaz existuje. V tomto SlashCommandBuilderu lze nastavit jméno a popis a to zda-li má více pod-příkazů, to znamená více argumentů v příkazu.



Obrázek 5: Další argumenty příkazů

module.exports = {

data: **new** SlashCommandBuilder()

.setName('message')

.setDescription('message specific player on server')

.addStringOption(option => option.setName('target').setDescription('target to message'))

Zdrojový kód č. 4: Vytvoření nového příkazu pomocí SlashCommandBuilderu

Názvy příkazů musejí být s malými písmeny, bez diakritiky a jedno slovo pro každý argument, jinak Discord příkaz nezaregistruje.

Po zadání příkazu se spustí funkce k němu přiřazená. V tomto případě je to pokaždé GET[[8]](#footnote-8) požadavek na server pomocí Axiosu. Spolu s GET požadavkem lze poslat i params[[9]](#footnote-9) ve kterém bot posílá argumenty z už zmíněných pod-příkazů.

axios.get(`${process.env.API}/message`, {

params: {

target: target,

}

Zdrojový kód č. 5: Axios GET požadavek s přidanými parametry

URL adresa se nachází v dotenv souboru pro jednoduché přepsání při změně adresy serveru.

Po odeslání požadavku server vrací buď status 200 Ok s příslušnou zprávou nebo chybnou hlášku.

### Jednotlivé příkazy

#### Get online players

Příkaz /getonlineplayers pošle požadavek pro spočítání a zobrazení všech hráčů přítomných na serveru.

#### Give all cookies

Příkaz /giveallcookie zažádá o přidělení každému hráči, který se je přítomen na serveru, předmět sušenka do inventáře.

#### Kill all

Příkaz /killall pošle požadavek o zabití všech online hráčů.

#### Starve all

Příkaz /starveall je příkaz, který pošle takový požadavek, aby se všem hráčům na se všem hráčům dal hunger bar[[10]](#footnote-10) na nulovou hodnotu.

#### Give cookie

Příkaz /givecookie je příkaz podobný /giveallcookie, ale v tomto případě musí uživatel na Discordu napsat jako další argument hráčovo jméno.

#### Kill

Příkaz /kill je příkaz fungující na stejném principu jako /killall, v tomto případě ale musí uživatel na Discordu napsat jako druhý argument příkazu hráčovo jméno.

#### Message

Příkaz /message je příkaz pro poslání zprávy z Discordu jako soukromou zprávu hráči na serveru. Uživatel píšící příkaz musí jako druhý argument zadat hráčovo jméno a jako třetí argument zprávu kterou mu chce poslat. Zpráva může obsahovat velká i malá písmena, čísla i diakritiku.

#### Starve

Příkaz /starve je příkaz fungující stejně jako /starveall, v tomto případě je musí uživatel píšící příkaz dopsat i hráče na serveru kterého chce vyhladovět.

### Zaregistrování příkazů na server

Pro zaregistrování příkazů na server slouží soubor deploy.js. Pro vložení příkazů na server musí mít bod při založení označenou kolonku “applications.commands”. V souboru deploy.js se nachází funkce pro zaregistrování příkazů. Pro správné fungování funkce se musí do parametrů vložit id bota(CLIENT), id serveru(GUILD) na Discordu kam by se měli příkazy zaregistrovat

rest.put(Routes.applicationGuildCommands(process.env.CLIENT, process.env.GUILD), { body: commands })

Zdrojový kód č. 6: Zaregistrování příkazu pouze na jeden server

Tímto způsobem lze příkazy registrovat kdykoliv, ale pouze na jeden server zároveň. Globální registraci příkazů lze spustit jednou za hodinu. Proto je předchozí způsob doporučen pro vývoj. Pro globální registraci se musí funkce upravit tímto způsobem:

rest.put(Routes.applicationCommands(clientId),{ body: commands });

Zdrojový kód č. 7: Globální registrace příkazu

Pro spuštění registrace slouží příkaz do konzole nacházející se ve složce bota node deploy.js. Při úspěšné registraci se vrátí zpráva “Successfully registered commands on server”.

# Minecraft plugin

Pro zajištění komunikace s Discord botem byl vytvořen Minecraft plugin, který se chová jako http server. Psán byl v jazyce Java a v Intellij Idea.

## Vytvoření a nasazení

Plugin byl vytvořen pomocí Minecraft development rozšíření ve zmíněném ide. Po vytvoření, se objeví defaultní třída s názvem projektu, ve které jsou dvě metody OnEnable, ve které je logika při spuštění pluginu, a on OnDisable, ve kterém je logika při vypnutí pluginu.

Pro nasazení projektu se musí plugin buildnout[[11]](#footnote-11). Po buildnutí se vytvoří ve složce “target” .jar soubor, který se může vložit do Minecraft serveru do složky “plugins”. Tím se zajistí fungování pluginu pro příští spuštění serveru.

## Komunikace s klientem

Pro vytvoření http serveru byl použit základní HttpServer zabudovaný v Javě. Na který se pak klient napojoval.

HttpServer server = HttpServer.create(**new** InetSocketAddress("localhost", 8001),0);

Zdrojový kód č. 8: Vytvoření http serveru

## Koncové body

Server má vytvořené koncové body pomocí createContext metody, pro každý jednotlivý příkaz. Po poslání požadavku na jakýkoliv koncový bod se spustí handler[[12]](#footnote-12) která přečte požadavek od klienta, pokud existují uloží parametry a odpoví klientovi a uzavře spojení. Parametry se předělávají pomocí funkce queryToMap do HashMapy ze které se v handlerech čtou.

server.createContext("/get-online-players", exchange ->

Zdrojový kód č. 9: Vytvoření koncového bodu

exchange.sendResponseHeaders(200, response.**length**());

exchange.getResponseBody().write(response.**toString**().getBytes());

Zdrojový kód č. 10: Posílání odpovědi na požadavek

exchange.getResponseBody().close();

Zdrojový kód č. 11: Uzavření spojení s klientem

Každý příkaz je vždy spojený s jednotlivými nebo se všemi hráči. Proto se v každém handleru vytvoří kolekce, do které se načtou všechny hráči momentálně přítomní na serveru. S kolekcí se v kódu dále pracuje. Jestli klient pošle v parametrech hráče, který neexistuje nebo není online na serveru vrátí server klientovi zprávu “No such player exist”.

Collection<? **extends** Player> player = Bukkit.getServer().getOnlinePlayers

Zdrojový kód č. 12: Zjištění online hráčů na serveru

### Jednotlivé koncové body

#### Give all cookie

Koncový bod /give-all-cookie slouží pro přidání sušenky do inventáře všem online hráčům na serveru. Klientovy vrací zprávu “Giving cookies to all players”.

**for** (Player p : player) {

ItemStack itemStack = **new** ItemStack(Material.COOKIE, 1, (**short**) 1);

p.getInventory().addItem(itemStack);

}

Zdrojový kód č. 13: Vložení sušenky všem hráčům

#### Give player cookies

Koncový bod /give-player-cookie slouží pro přidání sušenky do inventáře určitému hráči. Hráč se zjistí z parametru “target” který poslal klient. Klientovy vrací zprávu “Giving cookie to” + “target”.



Obrázek 6: Sušenka v inventáři

#### Kill all

Koncový bod /kill-all slouží pro zabití všech hráčů na serveru. Hráči se zabijí tím že se jejich životy nastaví na nulovou hodnotu. Klientovy vrací zprávu “Killing all players”.

p.setHealth(0);

Zdrojový kód č. 14: Setnutí životů na nulu

#### Kill

Koncový bod /kill slouží pro zabití jednoho určitého hráče. Hráče, který má být zabit pošle klient v parametrech jako “target”. Klientovy vrací zprávu “Killing” + “target”.

#### Message

Koncový bod /message slouží pro poslání zprávy nějakému hráči na serveru. Klient pošle požadavek s parametry, ve kterých se nachází jméno hráče jako “target”, autor jako “author”, a zpráva jako “message”. Autor a zpráva se hráči zobrazí vlevo dole v chatu. Klientovy vrací zprávu “Sending message to” + “target”.

Objects.requireNonNull(Bukkit.getPlayer(params.get("target"))).sendMessage(params.get("author")+ ": " +params.get("message"));

Zdrojový kód č. 15: Vypsání zprávy do hry



Obrázek 7: Zpráva z /message příkazu

#### Starve all

Koncový bod /starve-all slouží pro vyhladovění všech hráčů na serveru. Vyhladovění způsobí nastavením hunger baru na nulovou hodnotu. Klientovy vrací zprávu “Staring all players”.

p.setFoodLevel(0);

Zdrojový kód č. 16: Setnutí hunger baru na nulu

#### Starve

Koncový bod /starve slouží pro vyhladovění určitého hráče na serveru. Hráč se zjistí z poslaných parametrů jako “target” od klienta. Klientovy vrací zprávu “Starving” + “target”.



Obrázek 8: Hunger bar s nulovou hodnotou

#### Get online players

Koncový bod /get-online-players slouží pro získání všech momentálně přítomných hráčů. Hráče následně spočítá a vrací klientovi počet hráčů a jejich jména ve hře. Klientovy vrací zprávu “Getting online players”.

# Závěr

Vytváření práce bylo ze začátku trochu obtížné a v průběhu se plány hodně měnily. Jediné, co se neměnilo byl její smysl, a to propojit Discord se Minecraft serverem. Co se týče Discord bota, tak se původně měli psát příkazy jako normální zpráva s prefixem podle kterého by bot zjistil, že se mluví na něj. Nakonec se ale přešlo na normální lomené příkazy kvůli přehlednosti a jednoduchosti. Co se týče Minecraft pluginu, tak měl být http server pomocí jiné lepší knihovny. Ale kvůli nenáročnosti a menšímu počtu příkazů se přešlo na už zabudovaný http server v Javě. Tato práce pouze reprezentuje tu možnost že propojit Discord bota a Minecraft server lze. Tento projekt jde velice jednoduše rozšířit a přidat více funkcí, nejen pomocí lomených příkazů. Myslím, že mnoho komunit by takového bota ve svém Discord serveru ocenili a že by nebylo vůbec špatné tento nápad do budoucna rozšířit.

# Přílohy

## Seznam obrázků

[Obrázek 1: Psaní příkazu 8](#_Toc102290095)

[Obrázek 2: Chyba na straně aplikace 8](#_Toc102290096)

[Obrázek 3: Špatně zadaný příkaz 9](#_Toc102290097)

[Obrázek 4: Struktura příkazů 9](#_Toc102290098)

[Obrázek 5: Další argumenty příkazů 10](#_Toc102290099)

[Obrázek 6: Sušenka v inventáři 17](#_Toc102290100)

[Obrázek 7: Zpráva z /message příkazu 18](#_Toc102290101)

[Obrázek 8: Hunger bar s nulovou hodnotou 19](#_Toc102290102)

## Zdroje

*Discord* [online]. [cit. 2022-05-01]. Dostupné z: <https://discord.com/>

*Minecraft* [offline]. [cit. 2022-05-01]. Dostupné z: <https://www.minecraft.net/>

*Axios* [online]. [cit. 2022-05-01]. Dostupné z: <https://axios-http.com/>

*Nodemon* [online]. [cit. 2022-05-01]. Dostupné z: <https://www.npmjs.com/package/nodemon>

*DiscordJs* [online]. [cit. 2022-05-01]. Dostupné z: <https://discordjs.guide/>

*Http server java* [online]. [cit. 2022-05-01]. Dostupné z: <https://docs.oracle.com/javase/8/docs/jre/api/net/httpserver/spec/com/sun/net/httpserver/HttpServer.html>

*Minecraft plugin* [online]. [cit. 2022-05-01]. Dostupné z: <https://www.spigotmc.org/>

*Env* [online]. [cit. 2022-05-01]. Dostupné z: <https://www.npmjs.com/package/dotenv>

1. Discord [online]. [cit. 2022-05-01]. Dostupné z: https://discord.com/ [↑](#footnote-ref-1)
2. Minecraft - [online]. [cit. 2022-05-01]. Dostupné z: https://www.minecraft.net/ [↑](#footnote-ref-2)
3. Hostovat – server je vytvořen a poběží na vašem počítači [↑](#footnote-ref-3)
4. Plugin – aplikace třetí strany [↑](#footnote-ref-4)
5. Discord developer portal [online]. [cit. 2022-05-01]. Dostupné z: <https://discord.com/developers/applications> [↑](#footnote-ref-5)
6. Axios [online]. [cit. 2022-05-01]. Dostupné z: https://axios-http.com/ [↑](#footnote-ref-6)
7. SlashCommandBuilder – Třída pro vytváření příkazů – součást Discord.js balíčku [↑](#footnote-ref-7)
8. GET – požadavek o získání něčeho ze serveru [↑](#footnote-ref-8)
9. Params – doplňující informace pro GET požadavek [↑](#footnote-ref-9)
10. hunger bar- bar ukazující hráčovu potřebu jídla [↑](#footnote-ref-10)
11. Build – projekt se vytvoří a je schopný dát se do produkce. [↑](#footnote-ref-11)
12. Handler – funkce starající se o další průběh požadavku na koncový bod [↑](#footnote-ref-12)